Storage Performance Analyzer (SPA)

Software Documentation
Version: 1.0.1

SPA Development Team

Documentation Build Time: Thursday 29th January, 2015 at 13:13

Contents

1. Introduction

1.1 About

This document will explain how the Storage Performance Analyzer (SPA) can be used for benchmarking. This guide starts with a high-level introduction of the architecture. Then, it includes the steps that are needed to set up the system under test as well as a step-by-step guide how to compile the tool, configure the benchmark that should be run, the actual benchmark, and the data retrieval using the R libraries. This document concludes with a minimal running example with screenshots and a Q&A section.

The web page of SPA is available at `http://storageperformanceanalyzer.github.io/SPA/`. Publications describing the tool is available at [NBKR13, NBR$^+$14].

1.2 Requirements

Controller Machine:

- Platform independent (Windows, Unix and Mac operating systems tested)
- Java 6 or later, Apache Ant for compilation
- R version 3 or later (r-project.org)
- SQLite version 3 (sqlite.org)

System Under Test (SUT):

- POSIX compatible operating system (tested on Linux)
- SSH access from the controller machine

1.3 Download

Platform-independent *source code* as well as *prepared drops* (i.e., already generated model code) for common operating systems both including *examples* can be downloaded from the project web page `http://storageperformanceanalyzer.github.io/SPA/`.

2. Architecture

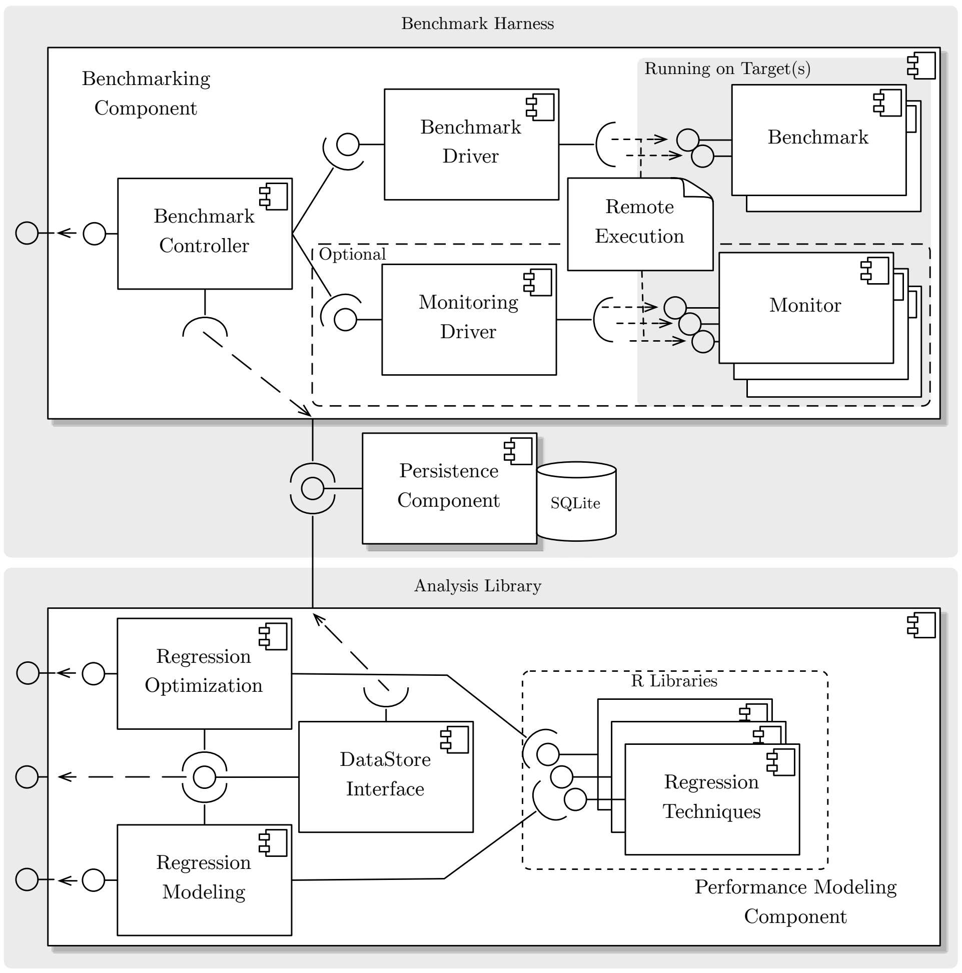


Figure 2.1: High-level Overview and Components of SPA

Figure 2.1 gives an high-level overview of SPA. The tool basically consists of a *Benchmark Harness* and an *Analysis Library*. The Benchmark Harness, which runs on a master machine, controls the parallel execution of benchmarks running on one or more remote targets (e.g., virtual machines). Furthermore, the benchmarking targets can be monitored using several self-composed or operating system monitors, e.g., *blktrace*, where its *Monitor Driver* already exists. The Analysis Library can be used to retrieve the measurements and use modeling functions to analyze the data and create regression models.

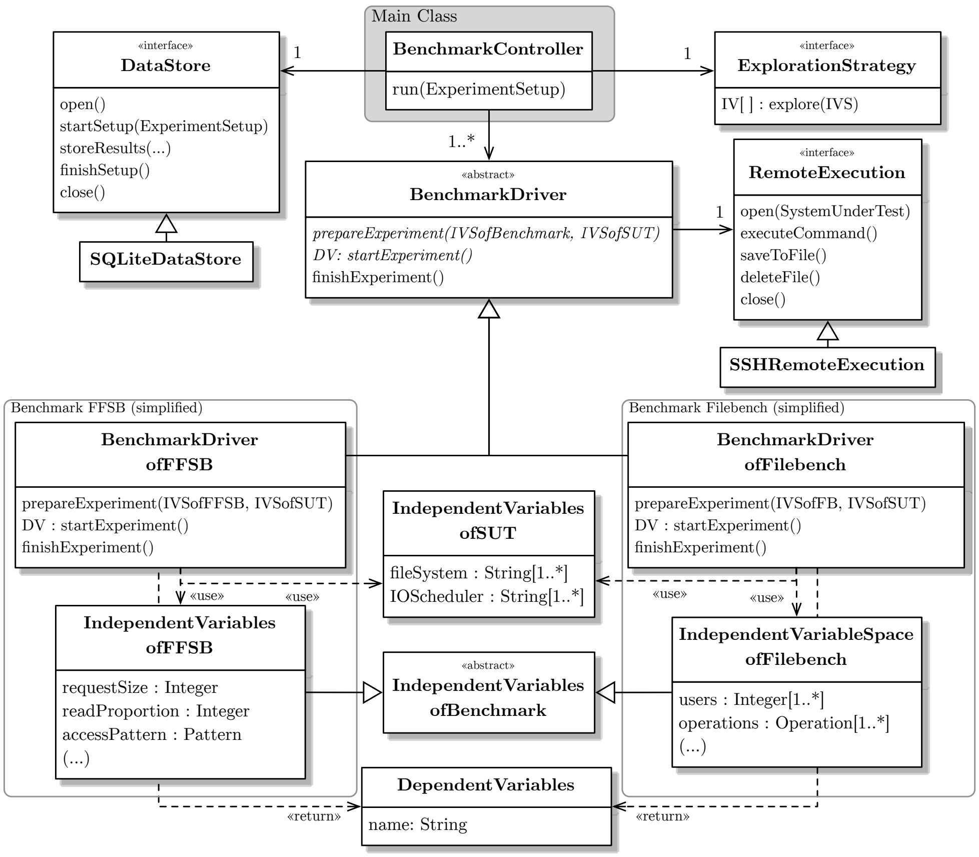


Figure 2.2: Most Important Classes of the Benchmark Harness

The most important classes of the Benchmark Harness are shown in Figure 2.2. The *Benchmark Controller* is the main and core class. Also shown are how two specific benchmarks are integrated in the tool using a *Benchmark Driver* and the respective configuration or *Independent Variables*.

A typical benchmarking sequence with two targets (represented by the Benchmark Drivers) is shown in Figure 2.3. The Benchmark Controller sets up the datastore and for every benchmark configuration element as explored by the *Exploration Strategy* the measurements are started. The benchmark (or experiment) is prepared, started, and then finished. The results are stored to the datastore. The coordination is multi-threaded for a parallel execution of the benchmark and asynchronous data persistence.

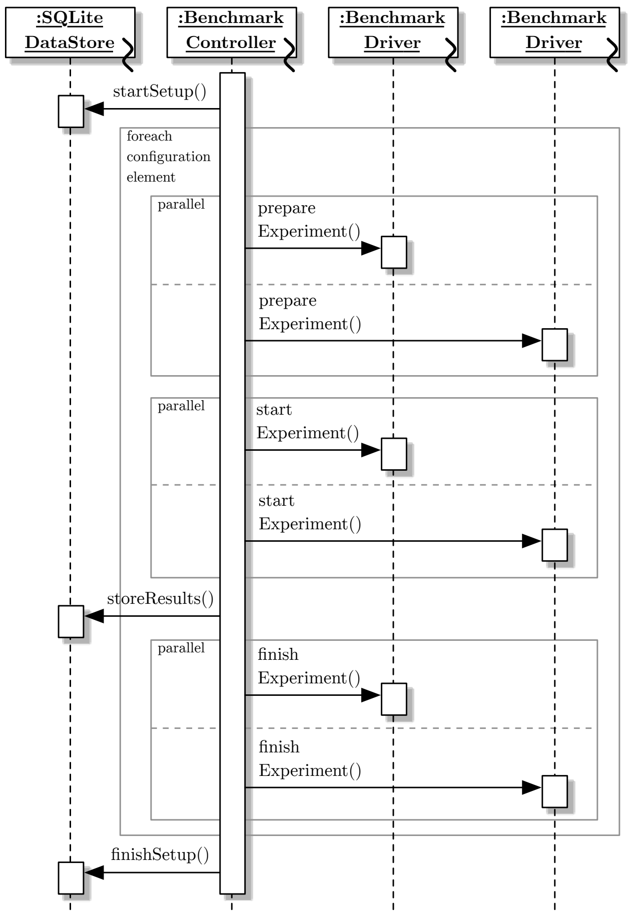


Figure 2.3: Typical Benchmarking Sequence

3. System Under Test Setup

Some manual steps are needed once on every system under test to set it up for use together with the storage benchmark harness. The tool expects some preconditions to be true. The following preconditions must be checked and met on all systems under test.

3.1 Authentication and Login

The Storage Benchmark Harness must be able to connect to the machine and log in using ssh. For this reason, a user which must have a shell and therefore be able to login must exist.

The authentication of the user is done using ssh keys. This method is a alternative to ordinary passwords. It uses a public/private key pair where the private key is only held by the client and the public key is stored on all servers. Every client possessing the private key can connect to every server where the public key is stored.

To generated a new key pair, the `ssh-keygen` application can be used. This application is included in the default OpenSSH[1] distribution package. During the creation of the key, no password may be specified. The Storage Benchmark Harness relies on a key without a password because it has no abilities to ask for a password during benchmarking. If the user already has a public/private key pair, it can choose to either reuse this key for the benchmarking or generate a second pair. The full path of the key should be noted. The default path for the first key is typically `${HOME}/.ssh/id_rsa` but another path can be specified upon key creation.

After a new key has been created or an existing key has been selected for reuse, the key must be copied to all servers which will be involved in the benchmarking. The can be done using the `ssh-copyid` command. A typical call is `ssh-copyid -i ${HOME}/.ssh/id_rsa_bench username@hostname`. An alternative is to copy the file manually if `ssh-copyis` is not available per default (like on Mac OS for example): `scp ${HOME}/.ssh/id_rsa_bench.pub username@hostname:${HOME}/.ssh/authorized_keys`. During this copying the user has to specify the password which is set for the user on the remote host. This command must be repeated for all hosts which will be involved in the benchmarking.

To test whether the key generation has succeeded, the user can simply try to connect to the remote host by using `ssh username@hostname -i ${HOME}/.ssh/id_rsa_bench`. If the connection succeeds without entering a password, the key was setup successfully.

[1] `http://openssh.org/`

For further documentation on the public/private keys see the man-pages of ssh, ssh-keygen and ssh-copyid.

3.2 Benchmark and Monitor Installation

The benchmark must be installed on every system under test. Additionally its executable must be in a directory which is included in the users PATH variable. On a typical Linux installation, the ${HOME}/bin directory is already included in the path. If it is not, this can be achieved for one user by creating (or adjusting) the file .bash_profile with the following lines:

```
$ PATH=$PATH:${HOME}/bin
$ export PATH
```

If this has to be changed system wide (except *root*), those lines can be appended to the file /etc/profile. The benchmark binaries can then be symlinked in the ${HOME}/bin directory for easier maintenance.

The monitoring tool BLKTRACE can be used and needs to be installed using the Linux tool Aptitude.

3.2.1 Flexible File System Benchmark

For the FFSB Benchmark, the Storage Benchmark Harness needs an extended version. This version was forked from the original source and is available online[2]. To install FFSB and make it accessible for the Storage Benchmark Harness, the user can use the following procedure after having logged in on the system under test:

```
$ cd ${HOME}
$ svn co https://github.com/FFSB-Prime/ffsb
$ # alternatively using git:
$ # git clone https://github.com/FFSB-Prime/ffsb
$ cd ffsb/trunk
$ ./configure
$ make
$ ln -s  ${HOME}/ffsb/trunk/ffsb ${HOME}/bin/ffsb
```

The FFSB Benchmark driver must also modify files which are only writable by the superuser. To achieve this, it uses the sudo command. Because the Storage Benchmark Harness can not input passwords, the commands which need to be executed using sudo must not ask for a password. Currently the only tool which the storage benchmark harness must execute using sudo is the tee command. In specific, the command is used to set or change the I/O scheduler as specified in the experiment configuration. To make the execution of this command password-less, edit the *sudoers* file using sudo visudo and append the following line:

```
ffsbTest ALL=NOPASSWD:/usr/bin/tee
```

Change ffsbTest to the appropriate user.

As for all storage benchmarks, the FFSB benchmarks needs a target directory where the actual benchmarking should happen on the machine. Per default FFSB used the /tmp/ffsbtarget directory on the system under test as target directory. This directory was choosen because /tmp is writable by all users. In this way a symlink can be used to select the actual target for ffsb. For example, if the /mnt/huge/target directory should be used, the following command creates the right symlink:

[2]https://github.com/FFSB-Prime/ffsb

```
$ ln -s /mnt/huge/target /tmp/ffsbtarget
```

If for some reasons the directory which FFSB uses as target should be set directly, this can be done by using the `ffsbtargetdir` environment variable.

The installation of the benchmark, the modification of the *sudoers* file and the symlinking of the target directory must be repeated for all systems under test where a ffsb benchmark experiments should be executed.

3.2.2 Filebench

A bigfixed version of Filebench is available online[3] and can be checked out using git. The user can take the following command to download the source code from the git-repository and install the benchmark:

```
$ cd ${HOME}
$ git clone https://github.com/Filebench-Revise/Filebench-
  Revise.git filebench
$ cd filebench
$ ./configure
$ make
$ ln -s  ${HOME}/filebench ${HOME}/bin/filebench
```

As described in the FFSB Section the Filebench needs a directory to store its filesets. Filebench's default fileset location is set to the `/tmp/filebenchtarget` directory on the system under test. Similar to FFSB a symlink can be used to point to a different location (for example to `/mnt/huge/target`):

```
$ ln -s /mnt/huge/target /tmp/filebenchtarget
```

If for some reasons the directory which Filebench uses as target should be set individually, this can be done by using the `filebenchtargetdir` environment variable. Since filebench is restarted in case of an error, the following root rights are necessary, `echo` to obtain the process ID and `kill` to stop the process:

```
user ALL=NOPASSWD:/bin/echo
user ALL=NOPASSWD:/bin/kill
```

3.2.3 Blktrace

The Blktrace monitoring tool is usually part of the `sysstat` package and can be installed using the bash command:

```
$ apt-get update && apt-get install sysstat
$ # sudo apt-get update && sudo apt-get install sysstat
```

The installation of the `sysstat` package requires root access. On some distributions, Blktrace has its own package, in that case replace `sysstat` with `blktrace` in the command.

Per default Blktrace records its results to the `/tmp/blktracetarget` directory on the system under test. As before the location can be changed using a symlink:

```
$ ln -s /mnt/huge/target /tmp/blktracetarget
```

Alternatively it can be set by the help of the environment variable `blktracetargetdir`.

Blktrace only runs with root rights. To post-process the monitoring files without root rights, the owner of the files created by Blktrace need to be changed to a normal user.

[3] `https://github.com/Filebench-Revise`

This is achieved using the `changeOwner.sh` script that modifies the owner of the files to the owner of the respective parent folder. This avoids requiring root right for the `chown` command. The `changeOwner.sh` script needs to be placed correctly and the execution attribute has to be set:

```
   $ /usr/local/bin/changeOwner.sh
   $ chmod u+x /usr/local/bin/changeOwner.sh
```

Storage Benchmark Harness needs further root rights to allow a correct processing of all recording and analyzing steps. Thus, the sudoers file has to be extended by the following lines:

```
user ALL=NOPASSWD:/usr/sbin/blktrace
user ALL=NOPASSWD:/usr/local/bin/changeOwner.sh
user ALL=NOPASSWD:/bin/echo
user ALL=NOPASSWD:/bin/kill
```

Change `user` to the appropriate user. It is strongly recommended to use `sudo visudo` to modify the sudoers file.

4. Installation and Compilation

The Storage Benchmark Harness must be installed only on the measurement machine. It needs an installed Java 6 JRE for running and a Java 6 SDK for compilation. Additionally, Apache Ant[1] must be installed as build tool.

The source of the tool consists of two projects: The EMF[2] model (*StorageBenchmarkHarnessModel*) and the actual tool (*StorageBenchmarkHarness*). When first using the tool, the two projects must be imported into an Eclipse with EMF Tools installed, e.g., using Eclipse Modeling Tools[3]. The code of the *StorageBenchmarkHarnessModel* needs to be generated by using `SBHModel.genmodel` in the `model` folder. Simply open the file inside Eclipse and right-click on the `SBHModel` package and choose `Generate All`. The source can then be compiled by switching into the `StorageBenchmarkHarness` folder and executing `ant`, e.g., using the shell/terminal. As all libraries are included in the projects, the compilation should work without further interaction. After this, the `run` binary in the folder should work and display a help screen when executed using the shell[4]. Alternatively, the `main` method of `BenchmarkController` can be run inside Eclipse and the help message is shown on the console. Parameters can then be passed over the `run` configuration.

[1]`http://ant.apache.org/`
[2]Eclipse Modeling Framework
[3]`http://www.eclipse.org/downloads/packages/eclipse-modeling-tools/junosr1`
[4]Tested on UNIX-based systems.

5. Configuration

After installing the tool, the next sections explain its configuration. For the configuration of the tool, multiple pieces of information have to be specified:

- **System under test**: The connection information for each system under test must be configured. This includes the hostname, a user which can be used for logging and and a ssh-key which can be used for authentication.
- **Experiments**: The experiments which should be run on the hosts must be configured. This includes defining the independent variables by specifying independent variable spaces.
- **Experiment series**: The experiment series make the connection between a system under test and an independent variable spaces. It therefore defines which experiments should be executed on a system under test.
- **Experiment setup**: The experiment setup contains multiple experiment series. In this way it contains the specification which experiments should be run on which hosts (In contrast to the experiment series which specifies which experiment should be run on *one* host).

These information can be either stored in a single configuration file or distributed over multiple files for simpler reuse. The Storage Benchmark Harness uses the Eclipse EMF Framework for the configuration file generation, editing and parsing. The configuration files can be either edited by hand or using the Eclipse EMF Tools.

5.1 Eclipse Setup

To use the EMF Tools, first Eclipse[1] has to be set up. The EMF Tools have to be installed together with eclipse either by download a package which already includes the EMF or by later installing them.

The two projects which form the Storage Benchmark Harness can be imported into eclipse as they are eclipse projects. This can be done by right-clicking in the *Package Explorer*, selecting Import, *Existing Projects into Workspace.* In the dialog the import can be finished by selecting the parent directory (the directory containing the `StorageBenchmarkHarness` and `StorageBenchmarkHarnessModel` directories). After the import as well as the code generation described in Section 4, the eclipse errors should go away after some seconds.

[1]`http://eclipse.org`

5.2 Configuration File Creation

To create a new configuration file using the EMF Tools, launch an eclipse instance in your eclipse or, more conveniently, install the plugins of the EMF model into your host eclipse. To do the latter, do the following (see `http://help.eclipse.org/juno/index.jsp?topic=%2Forg.eclipse.pde.doc.user%2Ftasks%2Fui_export_install_into_host.htm`):

1. Open the export wizard, either Open the plugin export wizard *File > Export... > Plug-in Development > Deployable plug-ins and fragments*
2. Select the plug-ins to export and install
3. Select the last option on the Destination tab Install into host. Repository. Then choose a directory to create the repository in
4. Hit Finish. The export operation will run followed by the installation operation.
5. If the operations completed successfully, you will be prompted to restart. Choose to restart now
6. Your plug-ins will be installed and running after the restart. You can see what has been installed using the Installation Details button on the About Dialog (available by going to Help > About Eclipse SDK)

You will now be able to create instances of the configuration model using the wizard. Use *File > New > Other... > Example EMF Model Creation Wizards > Configuration Model.* Choose destination and set *Model Object* to *Configuration.*

As an *alternative*, the following steps are possible as well *in theory*:

1. Open the EMF Model in the `StorageBenchmarkHarnessModel/model/SBHModel-.ecore` file using the *ECore Model Editor.*
2. Right click on the *Configuration* EClass in the *SBHModel/Configuration* package.
3. Select *Create Dynamic Instance.*
4. Choose the filename and path for the configuration file and click on *Finish.*
5. Open the newly created configuration file using the *Reflective ECore Model Editor.* Due to a bug in EMF, the file get opened by the wrong editor by default.

5.3 Referencing Configuration Fragments

If the configuration is split in multiple configuration files, one configuration file must reference instances from another configuration file. To include the instances from other configuration files in the currently opened file, select *Reflective Editor* from the main menu of eclipse and then use the *Load Resource* functionality.

5.4 Runnable Configuration Files

The configuration model is shown in Figure 5.1. Configuration files which will be used for the Storage Benchmark Harness must include a *Experiment Setup* child. This child contains all the information which specifies multiple benchmark runs on multiple systems under test. The actual specification can either be placed in the same configuration file or be included from other files.

The *Experiment Setup* has the following configuration options:

- *Identifier*: A string which can be contain any text. This text is saved together with the results and can be very helpful in identifying the configuration.

- *Repeat Count*: A number which specifies how ofter each experiment should be repeated. Only positive numbers are allowed.

The only children which a *Experiment Setup* can contain are *Experiment Series*. A single experiment series specifies which experiments should be run a host. Therefore an experiment series is needed for every host which is involved in the benchmarking. Aside from the *Identifier* which is only used in the configuration file itself, the *Experiment Series* contains the following properties. All of them are references and can therefor point to structures from other files. The instances must already exist to be selected.

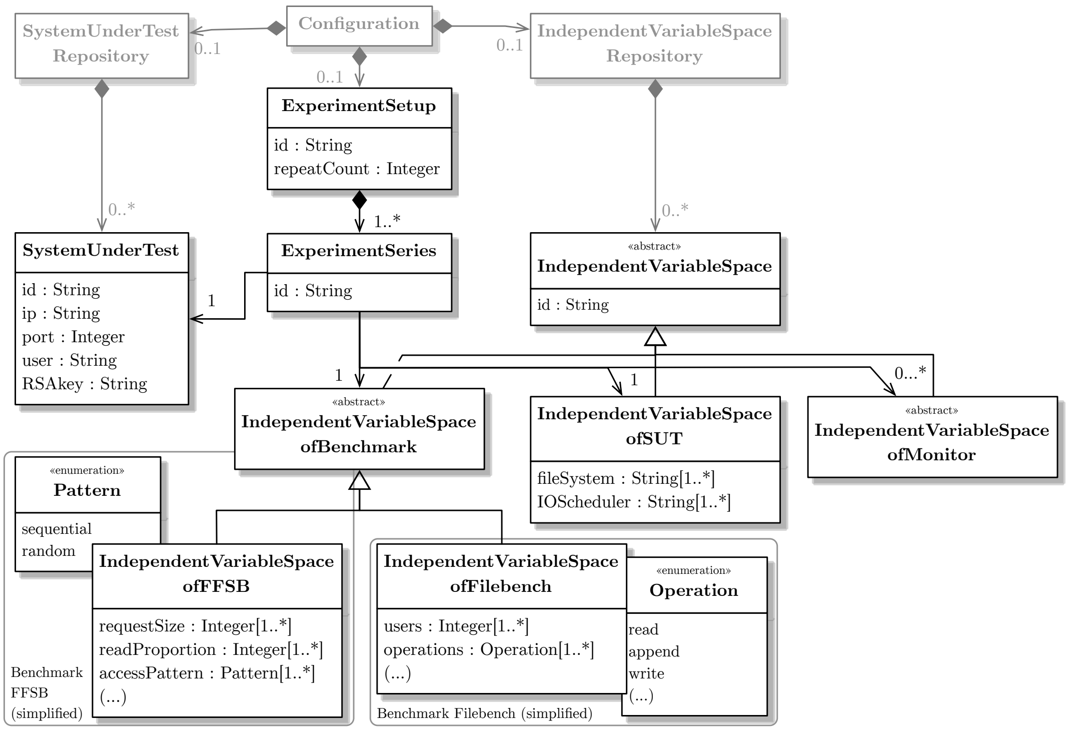
- *System Under Test*: This property links to a fragment containing all necessary information for connecting to a system under test. A new instance can be created as a child of a *System Under Test Repository*. This repository can be added to any *Configuration* instance, either the current configuration file or another file. The properties of the *System Under Test* are self-explanatory except for the *Key File*. This string must reference to the public key which was generated during the ssh private/public key generated explained above (see section 3.1).
- *Independent Variable Space of Sut*: The fragment to which this property points to contains values for all variables which are not specific for a benchmark but instead needed for every system under test. This currently includes the file system and the scheduler. Instances of this fragment can be created in the *Independent Variable Space Repository*. This repository again can be added to any *Configuration* instance.
- *Independent Variable Space of Benchmark*: This can point to a variable space for a specific benchmark. For example the *Independent Variable Space of FFSB* can be used if a FFSB Benchmark should be run on the host. This instance can also be added to the *Independent Variable Space Repository* (see above). Each variable space contains variables specific for a single benchmark.
- *Independent Variable Space of Monitor*: This optionally points to a variable space for a specific monitor tool. For example the *Independent Variable Space of Blktrace* can be used if the BLKTRACE monitoring tool should be used to monitor the corresponding benchmark run. Again the instance can be added to the *Independent Variable Space Repository*.

5.5 Benchmark Configurations

The configuration of a benchmark is defined in the respective `IndependentVariableSpace-ofBenchmark` instances. For full details, please refer to the help files of the benchmarks. It is important to note that the `IndependentVariableSpaceofBenchmark` instances do only allow to configure a subset of all possible configuration possibilities since the benchmarks themselves do not necessarily support all configurations they actually allow. The configurations are therefore limited to either the most interesting or the best tested ones. However, there is no inherent limitation such that the configurations and the `BenchmarkDriver`s can be extended for other parameters.

For creating new configurations, please study the examples provided with the tool and compare your configurations with the actual benchmarks using their raw file output. You can also use the `Validate` option of the Eclipse tree editor to identify configuration errors.

Hint: While the configuration units of Filebench can be passed, the units of FFSB configurations have been fixed for the sake of simplicity. They are *bytes* for `(read|write)BlockSize`, *kilobytes* for `fileSize`, and *megabytes* for `fileSetSize`. The FFSB `BlockSize` (request size) can be either specified for both read and write separately, or for both combined using the parameters without 'read' or 'write' pre-/postfix.



SystemUnderTest Repository
Configuration
IndependentVariableSpace Repository
0..1
0..1
0..1
ExperimentSetup
id : String
repeatCount : Integer
0..*
1..*
0..*
SystemUnderTest
id : String
ip : String
port : Integer
user : String
RSAkey : String
ExperimentSeries
id : String
1
«abstract»
IndependentVariableSpace
id : String
1
1
0...*
«abstract»
IndependentVariableSpace ofBenchmark
IndependentVariableSpace ofSUT
fileSystem : String[1..*]
IOScheduler : String[1..*]
«abstract»
IndependentVariableSpace ofMonitor
«enumeration»
Pattern
sequential
random
IndependentVariableSpace ofFFSB
requestSize : Integer[1..*]
readProportion : Integer[1..*]
accessPattern : Pattern[1..*]
(...)
Benchmark FFSB (simplified)
IndependentVariableSpace ofFilebench
users : Integer[1..*]
operations : Operation[1..*]
(...)
«enumeration»
Operation
read
append
write
(...)
Benchmark Filebench (simplified)

Figure 5.1: Configuration Model

6. Running the Benchmark

After the configuration process has been completed and the configuration file is finished, the benchmark can be started. This is done using the `run` command in the `StorageBenchmarkHarness` directory. This command should be executed in a terminal window as the application outputs all relevant information on the standard out. The `run` command needs the following parameters:

- `-c` or `-conf`: An absolute path to the configuration file which should be run. This configuration must contain a *Experiment Setup* as explained above.
- `-d` or `-database`: An absolute path to the database file which should be used to save the results. If the database does not exist, an empty database is created.

For additional parameters and their meaning see the output of the help screen or the source code.

6.1 Additional Output Files

The Storage Benchmark Harness automatically parses and transforms the output of the benchmarks into the database. If for debugging purposes or because additional processing should be done later one the raw output of the benchmarks should be saved, the `-r` or `-rawfilesavedir` parameter for the `run` command can be used. This parameter should contain a directory in which the raw files will be saved during the benchmarking. The user must pay attention to the additional space which is necessary to store these files.

6.2 Examples

An example of a full command to run an experiment setup would look similar to this:

```
filebenchtargetdir=/mnt/noorsh \
./run -c ./configs/BenchmarkConfiguration.xmi \
-d ./results/database.sqlite \
-r ./results_raw/ \
| tee -a ./logs/spa.log
```

The results are stored in an SQLite database. Many operating system provide native command line access, for example on Mac OS, the statement to connect to the database would look similar to this:

```
 $ sqlite3 ./results/database.sqlite
```

To show the available tables simply use the following command (more help is available at `http://www.sqlite.org/sqlite.html`):

```
sqlite> .tables;
```

An even more convenient access is provided by the R libraries.

7. R Libraries

7.1 Introduction

For the analysis of measurement results, a set of tailored libraries for the widely accepted statistics tool R[1][2] have been developed. The starting point is the file `SPA.r` that loads all other libraries and a few examples automatically. These files build upon other standard libraries that can be installed automatically in a batch by using `installPackages.r` (simply load it using `source("filename")`).

7.2 Interface

The following parts contain the main functions and are loaded automatically in `SPA.r`[3]. For more information, the functions are also commented directly in code.

- `DataStoreInterface.r`: Loads the measurement data from the database.
  - `getAllFFSBVars(db, metric=NULL, type=NULL)`
  - `getAllFilebenchVars(db, metric=NULL, type=NULL)`

    Get the measurement data from the database located at `db` (path). Optionally, filter for certain metrics or result types. This can be specified using the `SPAMETRICCONSTANTS` and `SPATYPECONSTANTS` constant containers, respectively.

- `RegressionOptimization.r`: Optimizes for given measurement data, in- and dependent variables the configuration for the regression techniques.
  - `optimizeTechniques(method, formula, data, ranges=NA, nSplits, nExplorations, trace=0, fold=NA, nIterations=15)`

    Optimizes the `method` ("rpart", "earth", ",m5", or "cubist") according to a `formula` using the given `data`. Optionally, the `ranges` of the method can be specified. Parameters of the optimization algorithm have the prefix "`n`", for details refer to [NBKR13]. The algorithm is deterministic if a `fold` for the cross-validation (objective function) is passed. Verbose output for `trace` = 1 or 2.

[1] `http://www.r-project.org/`
[2] `http://www.rstudio.com/`
[3] *Hint:* Both the GUI and command line version of R allow auto completion with single/double `TAB` key.

- `RegressionModeling.r`: Creates the regression models either with a given configuration or in batch for comparing the models.
  - `fitCertainModel(formula, data, method, parameter)`

    Use the `parameter`s obtained with `optimizeTechniques()` or specify others to create a model of the form given in `formula` with the `data` and the specific `method`.
  - `fitModels<-function(formula, data, methods=list())`

    Alternatively, create a set of models of the form given in `formula` with the `data` with different methods and parameters to compare them. Optionally, a list of `methods` can be passed to create more models.

7.3 Examples

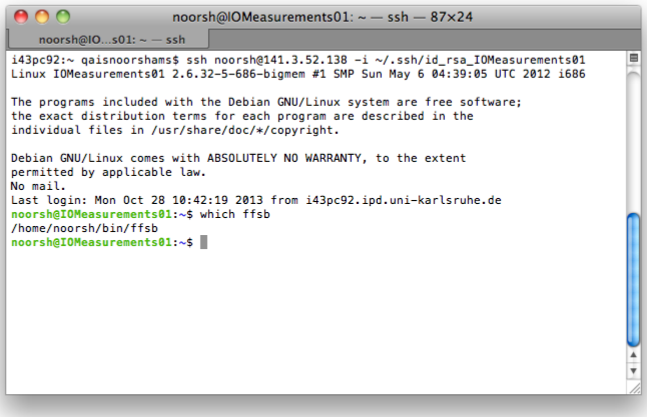
```
 > # you can use setwd(path) to switch the working directory
 > # Copy-paste of these lines will not work because of special
     symbols in the following such as quotes and tilde
 > source("installPackages.r")
 > source("SPA.r")
 > # The following example measurements are loaded automatically:
 > filebenchexample = getAllFilebenchVars("data/test/sqlite/
     filebenchtest.sqlite");
 > ffsbexample = getAllFFSBVars("data/test/sqlite/
     ffsbtest.sqlite", type=SPATYPECONSTANTS$mean);
 >
 > # example data provided by R (output abridged)
 > trees
    Girth Height Volume
 1:   8.3     70   10.3
 2:   8.6     65   10.3
 (...)
30:  18.0     80   51.0
31:  20.6     87   77.0
 >
 > # model with method EARTH
 > model = fitCertainModel(Volume~Girth+Height, trees, "earth",
     parameter=data.table("nprune"=3, "nk"=3,
     "degree"=1, "thresh"=0))
> summary(model, digits = 2, style = "pmax") # output abridged
Call: (...)
y =
  30
  + 6.6 * pmax(0, Girth -    14)
  - 3.5 * pmax(0,    14 - Girth)

Selected 3 of 3 terms, and 1 of 2 predictors
Importance: Girth, Height-unused
Number of terms at each degree of interaction: 1 2 (...)
GCV 14    RSS 313    GRSq 0.95    RSq 0.96
 > plotmo(model$finalModel)

 grid:    Girth Height
           12.9     76
 >
 > # another method: RPART
 > model = fitCertainModel(Volume~Girth+Height, trees, "rpart",
     parameter=data.table("cp"=0.01, "minsplit"=5))
 > plotCART(model$finalModel)
 >
 > # optimize the parameters
 > Eopt = optimizeTechniques(formula=Volume~Girth+Height,
     data=trees, method="rpart",nIterations=3, nExplorations=2,
     nSplits=3, trace=2)
 > modelOPT = fitCertainModel(Volume~Girth+Height, trees,
     "rpart", parameter=Eopt)
 > plotCART(modelOPT$finalModel)
 >
 > # Create a set of models to compare the techniques
 > models = fitModels(Volume~Girth+Height, trees)
     # (output abridged)
Creating Fold
(...)
Training M5.
Calculating resamples.
 > summary(models$compare) # (output abridged)
(...)
MAPE
                 Min. 1st Qu. Median   Mean 3rd Qu.  Max. NA's
lm              1.310   4.915  9.378 12.690  14.240 38.04    0
lm_2param_inter 2.651   5.794  6.215  7.408   9.744 13.75    0
(...)
m5              1.037   5.093  8.526 12.150  14.170 35.43    0
```

8. Minimal Running Example

In this example, User `noorsh` runs FFSB on machine `IOMeasurements01` using RSA key `id_rsa_IOMeasurements01`. The controller machine is Mac OS (shown briefly is the execution of the controller on a Debian Linux system), the measurement target is Debian Linux.

8.1 System Under Test Setup

As described in Section 3 and illustrated in Figure 8.1, ensure the user can connect to the system using RSA key without password prompt. And, as a minimal configuration, ensure FFSB is installed on the SUT and the binaries are in the user path, which can be checked using `which ffsb`. Root rights are only needed if the device scheduler should be modified.



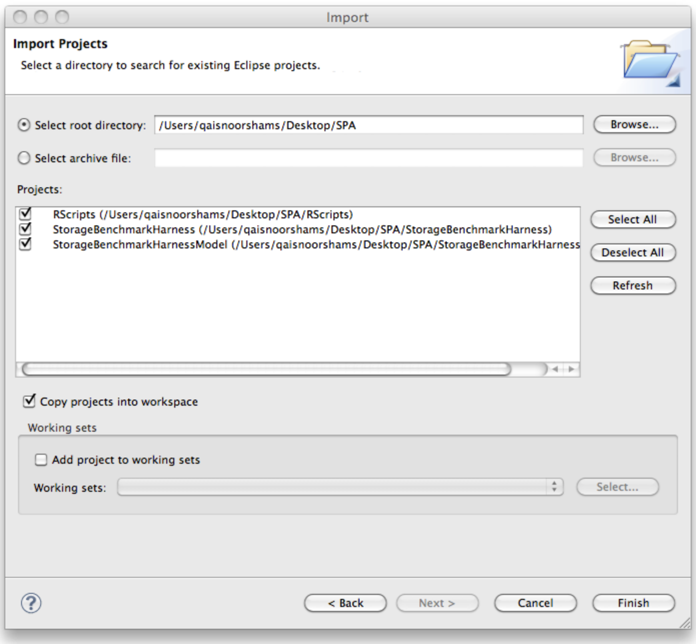
```
i43pc92:~ qaisnoorshams$ ssh noorsh@141.3.52.138 -i ~/.ssh/id_rsa_IOMeasurements01
Linux IOMeasurements01 2.6.32-5-686-bigmem #1 SMP Sun May 6 04:39:05 UTC 2012 i686

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
No mail.
Last login: Mon Oct 28 10:42:19 2013 from i43pc92.ipd.uni-karlsruhe.de
noorsh@IOMeasurements01:~$ which ffsb
/home/noorsh/bin/ffsb
noorsh@IOMeasurements01:~$
```

Figure 8.1: Connection to the SUT

8.2 Installation and Compilation

Download Eclipse Modeling Tools and import the SPA projects, cf. Figure 8.2.

After the import, you might get compilation errors and need to generate the EMF model code as shown in Figure 8.3 using `StorageBenchmarkHarnessModel/model/S-BHModel.genmodel -> package context menu -> Generate All` (skip this step if the



Import

Import Projects

Select a directory to search for existing Eclipse projects.

[✓ Select root directory:] /Users/qaisnoorshams/Desktop/SPA

Select archive file:

Projects:

- ✓ RScripts (/Users/qaisnoorshams/Desktop/SPA/RScripts)
- ✓ StorageBenchmarkHarness (/Users/qaisnoorshams/Desktop/SPA/StorageBenchmarkHarness)
- ✓ StorageBenchmarkHarnessModel (/Users/qaisnoorshams/Desktop/SPA/StorageBenchmarkHarness

✓ Copy projects into workspace

Working sets

☐ Add project to working sets

Working sets:

Figure 8.2: Import projects into Eclipse Modeling Tools

model code was already generated). Three new projects will be generated and all errors will disappear after a second if the project is rebuilt. In case the errors won't disappear, check the built path of the project `StorageBenchmarkHarnessModel` (`project context menu -> Properties -> Java Build Path`).

To easily manage the configuration files, install all plugins into the host as described in Section 5.2 and shown in Figure 8.4.

IMPORTANT: If the projects are installed into the host, Eclipse may delete the `build.xml` file in the `StorageBenchmarkHarnessModel` project. Simply save it and re-import the file if a manual rebuild is necessary.

You will now be able to open SPA configurations, e.g., located in `StorageBenchmarkHarness/configs/examples`, in the tree-based editor as shown in Figure 8.5.

You can execute the configurations using Eclipse run configuration or headless on the command line. For the former, open the `BenchmarkController` class in the package `edu.kit.sdq.storagebenchmarkharness` and click the run icon in order to execute the `main` method, cf. Figure 8.6. For the latter, navigate to the `StorageBenchmarkHarness` project and execute `ant` if the project was not yet built automatically by Eclipse. The project can be run using `./run`, cf. Figure 8.7. Either the Eclipse console or the command line will print the help.

IMPORTANT: If the projects were installed into the host, Eclipse may have deleted the `build.xml` file in the `StorageBenchmarkHarnessModel` project. Simply re-import the file if the project should be rebuilt manually.

NOTE: The code generation with Eclipse does only have to be done once. If this is done, the projects `StorageBenchmarkHarness` and `StorageBenchmarkHarnessModel` can be copied elsewhere and built and run headlessly, cf. Figure 8.8.

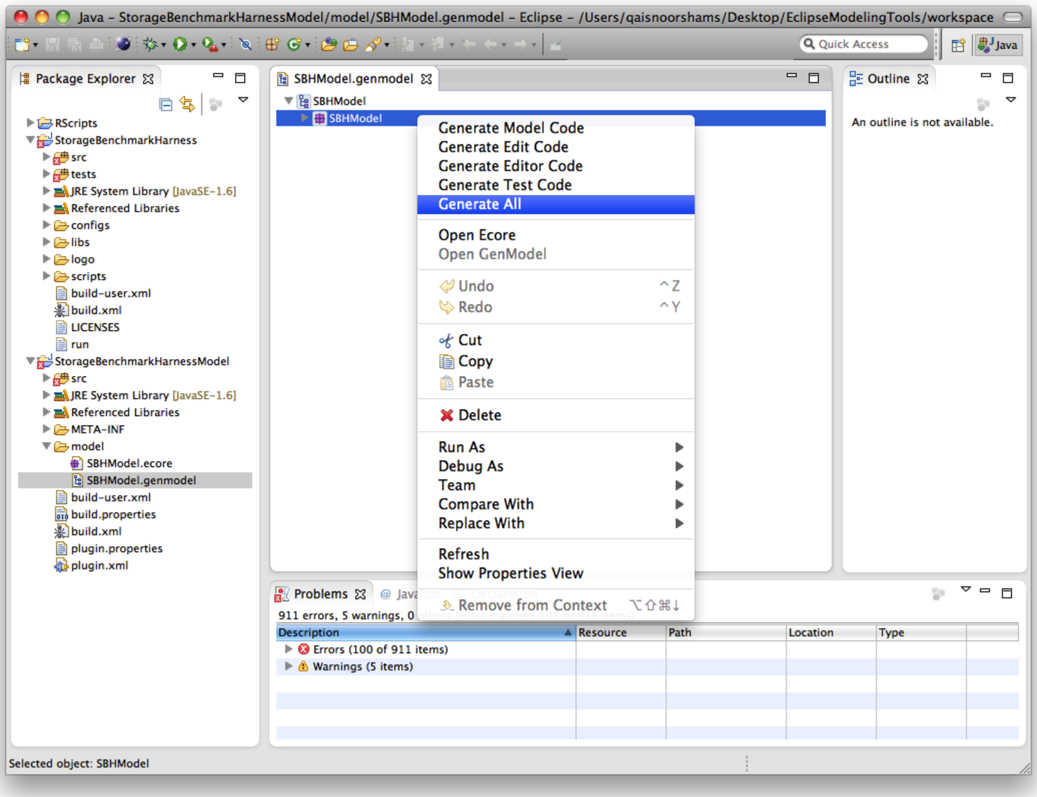


Figure 8.3: Generate EMF model code using a genmodel

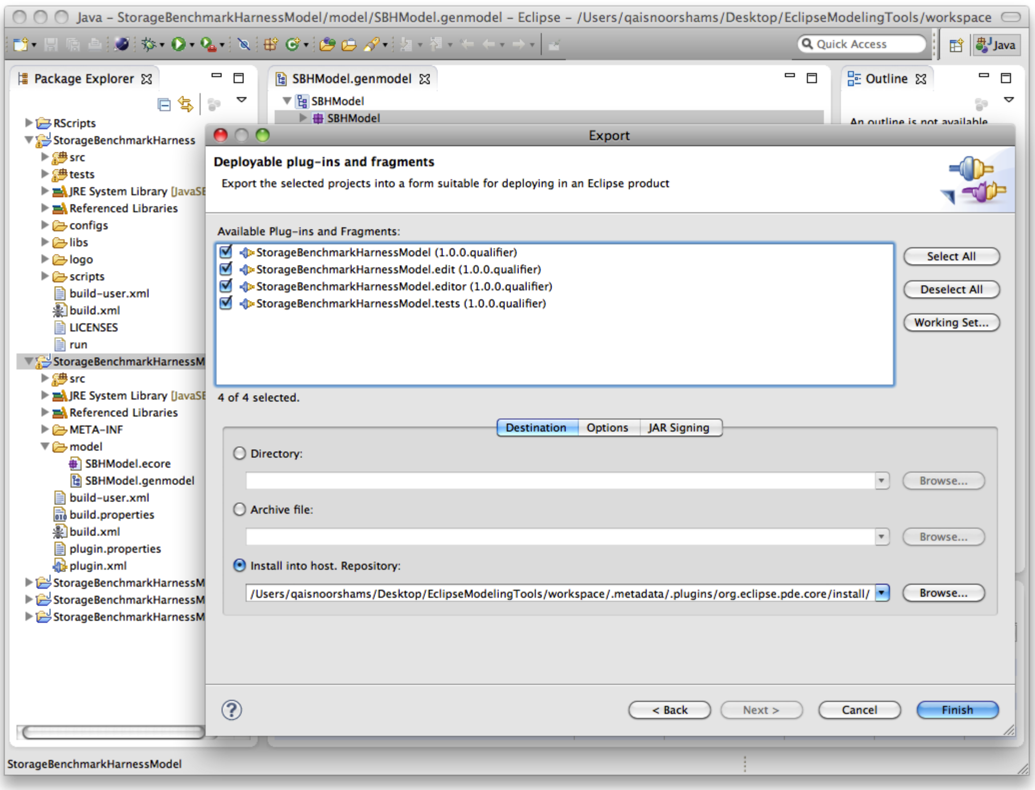
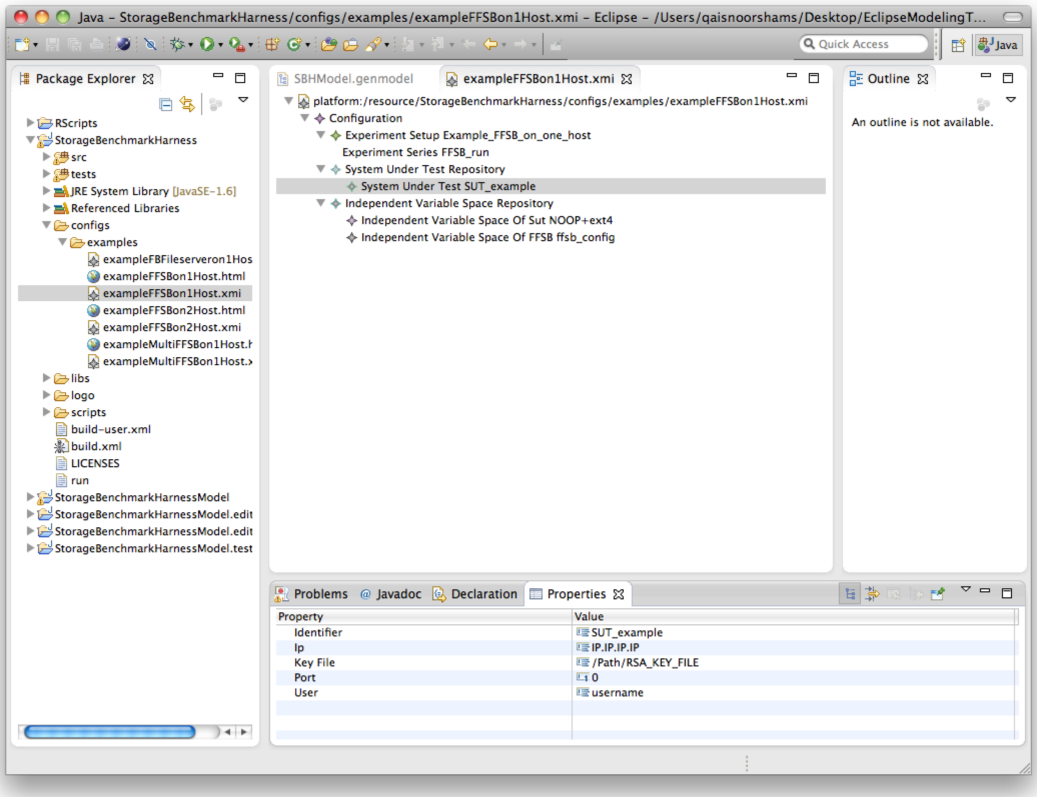

Java – StorageBenchmarkHarnessModel/model/SBHModel.genmodel – Eclipse – /Users/qaisnoorshams/Desktop/EclipseModelingTools/workspace
Quick Access
Java
Package Explorer
RScripts
StorageBenchmarkHarness
src
tests
JRE System Library [JavaSE
Referenced Libraries
configs
libs
logo
scripts
build-user.xml
build.xml
LICENSES
run
StorageBenchmarkHarnessM
src
JRE System Library [JavaSE
Referenced Libraries
META-INF
model
SBHModel.ecore
SBHModel.genmodel
build-user.xml
build.properties
build.xml
plugin.properties
plugin.xml
StorageBenchmarkHarnessM
StorageBenchmarkHarnessM
StorageBenchmarkHarnessM
SBHModel.genmodel
SBHModel
SBHModel
Outline
An outline is not available.
Export
Deployable plug-ins and fragments
Export the selected projects into a form suitable for deploying in an Eclipse product
Available Plug-ins and Fragments:
☑ StorageBenchmarkHarnessModel (1.0.0.qualifier)
☑ StorageBenchmarkHarnessModel.edit (1.0.0.qualifier)
☑ StorageBenchmarkHarnessModel.editor (1.0.0.qualifier)
☑ StorageBenchmarkHarnessModel.tests (1.0.0.qualifier)
Select All
Deselect All
Working Set...
4 of 4 selected.
Destination
Options
JAR Signing
Directory:
Browse...
Archive file:
Browse...
[✓ Install into host. Repository:]
/Users/qaisnoorshams/Desktop/EclipseModelingTools/workspace/.metadata/.plugins/org.eclipse.pde.core/install/
Browse...
< Back
Next >
Cancel
Finish
StorageBenchmarkHarnessModel

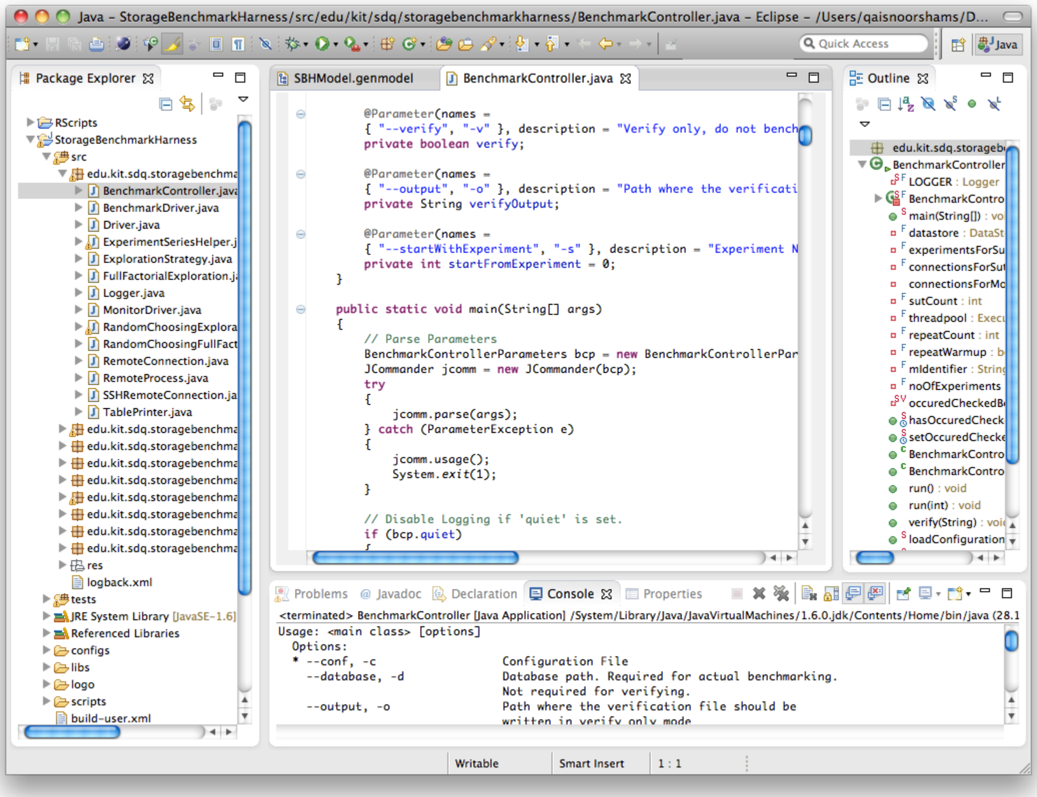
Figure 8.4: Install plugin projects into the host



```
platform:/resource/StorageBenchmarkHarness/configs/examples/exampleFFSBon1Host.xmi
  Configuration
    Experiment Setup Example_FFSB_on_one_host
      Experiment Series FFSB_run
    System Under Test Repository
      System Under Test SUT_example
    Independent Variable Space Repository
      Independent Variable Space Of Sut NOOP+ext4
      Independent Variable Space Of FFSB ffsb_config
```

Property	Value
Identifier	SUT_example
Ip	IP.IP.IP.IP
Key File	/Path/RSA_KEY_FILE
Port	0
User	username

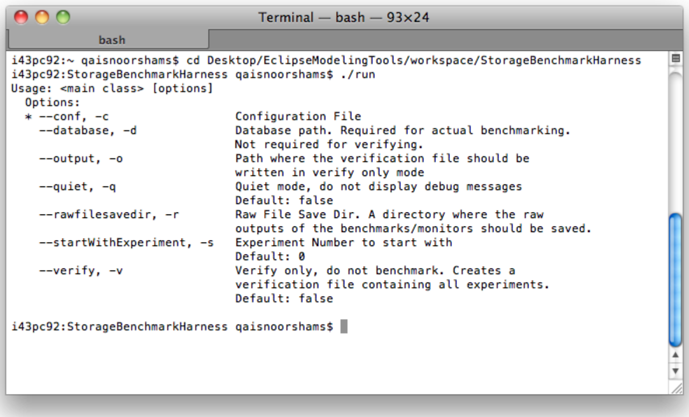
Figure 8.5: Example configuration in the tree-based editor



```
        @Parameter(names =
        { "--verify", "-v" }, description = "Verify only, do not bench
        private boolean verify;

        @Parameter(names =
        { "--output", "-o" }, description = "Path where the verificati
        private String verifyOutput;

        @Parameter(names =
        { "--startWithExperiment", "-s" }, description = "Experiment N
        private int startFromExperiment = 0;
    }

    public static void main(String[] args)
    {
        // Parse Parameters
        BenchmarkControllerParameters bcp = new BenchmarkControllerPar
        JCommander jcomm = new JCommander(bcp);
        try
        {
            jcomm.parse(args);
        } catch (ParameterException e)
        {
            jcomm.usage();
            System.exit(1);
        }

        // Disable Logging if 'quiet' is set.
        if (bcp.quiet)
```

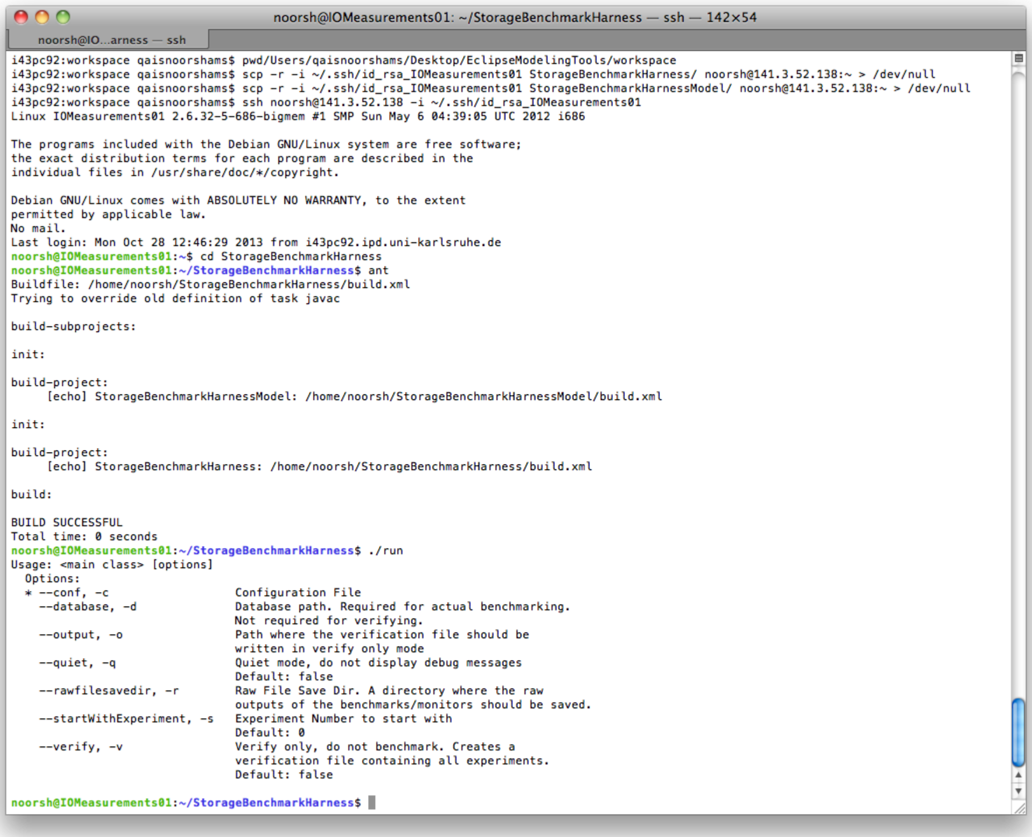
```
<terminated> BenchmarkController [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (28.1
Usage: <main class> [options]
  Options:
  * --conf, -c                   Configuration File
    --database, -d               Database path. Required for actual benchmarking.
                                 Not required for verifying.
    --output, -o                 Path where the verification file should be
                                 written in verify only mode
```

Figure 8.6: Run SPA in Eclipse



```
i43pc92:~ qaisnoorshams$ cd Desktop/EclipseModelingTools/workspace/StorageBenchmarkHarness
i43pc92:StorageBenchmarkHarness qaisnoorshams$ ./run
Usage: <main class> [options]
  Options:
  * --conf, -c                  Configuration File
    --database, -d              Database path. Required for actual benchmarking.
                                Not required for verifying.
    --output, -o                Path where the verification file should be
                                written in verify only mode
    --quiet, -q                 Quiet mode, do not display debug messages
                                Default: false
    --rawfilesavedir, -r        Raw File Save Dir. A directory where the raw
                                outputs of the benchmarks/monitors should be saved.
    --startWithExperiment, -s   Experiment Number to start with
                                Default: 0
    --verify, -v                Verify only, do not benchmark. Creates a
                                verification file containing all experiments.
                                Default: false

i43pc92:StorageBenchmarkHarness qaisnoorshams$
```

Figure 8.7: Run SPA in the terminal

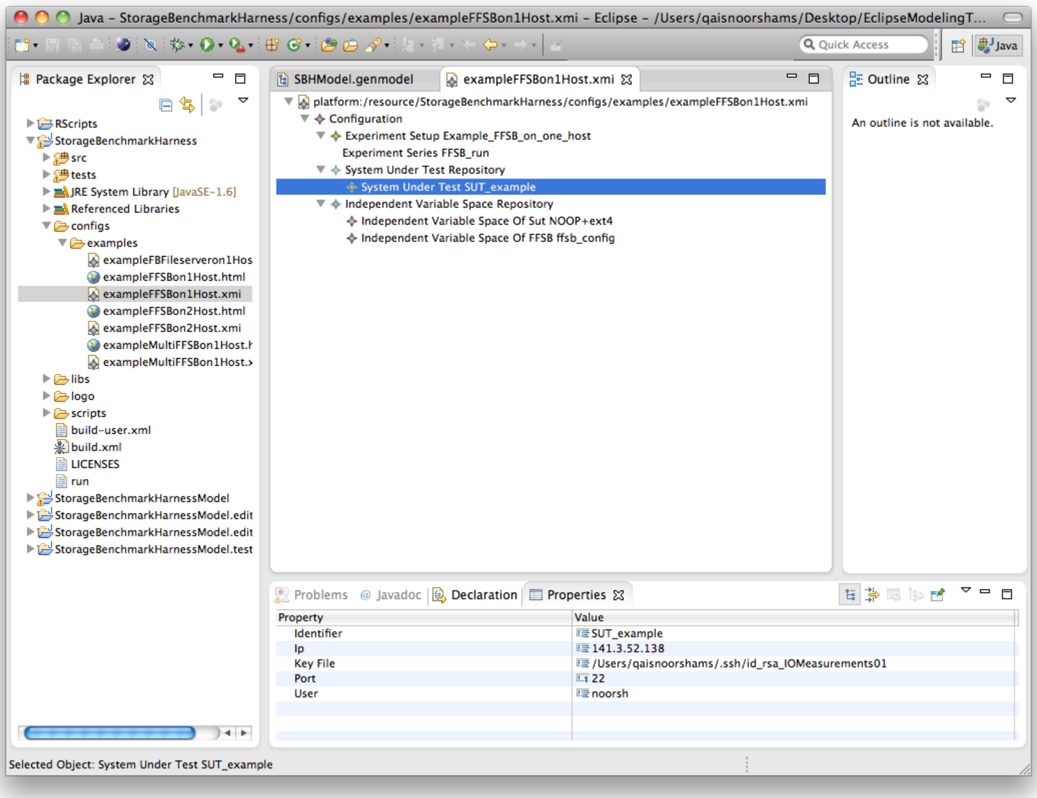

noorsh@IOMeasurements01: ~/StorageBenchmarkHarness — ssh — 142×54
noorsh@IO...arness — ssh

```
i43pc92:workspace qaisnoorshams$ pwd/Users/qaisnoorshams/Desktop/EclipseModelingTools/workspace
i43pc92:workspace qaisnoorshams$ scp -r -i ~/.ssh/id_rsa_IOMeasurements01 StorageBenchmarkHarness/ noorsh@141.3.52.138:~ > /dev/null
i43pc92:workspace qaisnoorshams$ scp -r -i ~/.ssh/id_rsa_IOMeasurements01 StorageBenchmarkHarnessModel/ noorsh@141.3.52.138:~ > /dev/null
i43pc92:workspace qaisnoorshams$ ssh noorsh@141.3.52.138 -i ~/.ssh/id_rsa_IOMeasurements01
Linux IOMeasurements01 2.6.32-5-686-bigmem #1 SMP Sun May 6 04:39:05 UTC 2012 i686

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
No mail.
Last login: Mon Oct 28 12:46:29 2013 from i43pc92.ipd.uni-karlsruhe.de
noorsh@IOMeasurements01:~$ cd StorageBenchmarkHarness
noorsh@IOMeasurements01:~/StorageBenchmarkHarness$ ant
Buildfile: /home/noorsh/StorageBenchmarkHarness/build.xml
Trying to override old definition of task javac

build-subprojects:

init:

build-project:
     [echo] StorageBenchmarkHarnessModel: /home/noorsh/StorageBenchmarkHarnessModel/build.xml

init:

build-project:
     [echo] StorageBenchmarkHarness: /home/noorsh/StorageBenchmarkHarness/build.xml

build:

BUILD SUCCESSFUL
Total time: 0 seconds
noorsh@IOMeasurements01:~/StorageBenchmarkHarness$ ./run
Usage: <main class> [options]
  Options:
  * --conf, -c                  Configuration File
    --database, -d              Database path. Required for actual benchmarking.
                                Not required for verifying.
    --output, -o                Path where the verification file should be
                                written in verify only mode
    --quiet, -q                 Quiet mode, do not display debug messages
                                Default: false
    --rawfilesavedir, -r        Raw File Save Dir. A directory where the raw
                                outputs of the benchmarks/monitors should be saved.
    --startWithExperiment, -s   Experiment Number to start with
                                Default: 0
    --verify, -v                Verify only, do not benchmark. Creates a
                                verification file containing all experiments.
                                Default: false

noorsh@IOMeasurements01:~/StorageBenchmarkHarness$
```

Figure 8.8: Build and run SPA remotely

8.3 Configuration

SPA configurations can be created using the tree-based editor of Eclipse. Eclipse can also check if a configuration is correct using the `validate` in the context menu. In the example, modify the `System Under Test` configuration of `StorageBenchmarkHarness/configs/examples/exampleFFSBon1Host.xmi` using the properties view. If the properties view is not shown by default, right-click on a given configuration element and choose `Show Properties View`. Change the SUT configuration according to your setup. In the example, the configuration looks as shown in Figure 8.9 (note that the key file is the *private key*). You can check the configuration using the root element `Configuration -> right-click -> Validate`, which should give you no error.


Java - StorageBenchmarkHarness/configs/examples/exampleFFSBon1Host.xmi - Eclipse - /Users/qaisnoorshams/Desktop/EclipseModelingT...
Quick Access
Java
Package Explorer
SBHModel.genmodel
exampleFFSBon1Host.xmi
Outline
Problems
@ Javadoc
Declaration
Properties

```
RScripts
StorageBenchmarkHarness
  src
  tests
  JRE System Library [JavaSE-1.6]
  Referenced Libraries
  configs
    examples
      exampleFBFileserveron1Hos
      exampleFFSBon1Host.html
      exampleFFSBon1Host.xmi
      exampleFFSBon2Host.html
      exampleFFSBon2Host.xmi
      exampleMultiFFSBon1Host.h
      exampleMultiFFSBon1Host.x
  libs
  logo
  scripts
  build-user.xml
  build.xml
  LICENSES
  run
StorageBenchmarkHarnessModel
StorageBenchmarkHarnessModel.edit
StorageBenchmarkHarnessModel.edit
StorageBenchmarkHarnessModel.test
```

```
platform:/resource/StorageBenchmarkHarness/configs/examples/exampleFFSBon1Host.xmi
  Configuration
    Experiment Setup Example_FFSB_on_one_host
      Experiment Series FFSB_run
    System Under Test Repository
      System Under Test SUT_example
    Independent Variable Space Repository
      Independent Variable Space Of Sut NOOP+ext4
      Independent Variable Space Of FFSB ffsb_config
```

An outline is not available.

Property	Value
Identifier	SUT_example
Ip	141.3.52.138
Key File	/Users/qaisnoorshams/.ssh/id_rsa_IOMeasurements01
Port	22
User	noorsh

Selected Object: System Under Test SUT_example

Figure 8.9: SUT configuration in the example

8.4 Running the Benchmark

As described above, the benchmark can be compiled and executed using Eclipse or the command line. Using the Eclipse run configuration, you set the program arguments using the `Arguments` tab and set the environment variables in the `Environment` tab. Using the command line, you can set environment variables preceding `./run` and program arguments after. Both methods are the same, so in the examples we will only show the usage in the command line. To see what the example configuration would benchmark, on the command line use `./run -c configs/examples/exampleFFSBon1Host.xmi -v -o exampleFFSBon1Host.html`. Open the newly created file `exampleFFSBon1Host.html` to see the table contains 1 line and 1 column `SUT_example` meaning that 1 experiment will be executed on the host `SUT_example`.

To run the configuration, use `ffsbtargetdir=/mnt/noorsh/ffsb/ ./run -c configs/examples/exampleFFSBon1Host.xmi -d exampleFFSBon1Host.sqlite -r raw|tee exampleFFSBon1Host.log` to benchmark at the location `/mnt/noorsh/ffsb/` of the SUT using the configuration `exampleFFSBon1Host.xmi` and save the results in the database `exampleFFSBon1Host.sqlite` as well as the raw files in the folder `raw`. Additionally, save the run output into `exampleFFSBon1Host.log`.

IMPORTANT: Make sure the scheduler of the target device is the same as specified in the configuration file in the `Independent Variable Space of SUT`, which is `NOOP` in the example, cf. Figure 8.9. The scheduler can be checked using `cat /sys/block/[DEVICE]/queue/scheduler`.

The log of the benchmark in the raw folder will look like this (abridged):

```
FFSB version 6.0-RC2~fork started

benchmark time = 60
ThreadGroup 0
================
         num_threads      = 10

         read_random      = on
         read_size        = 1048576      (1MB)
         read_blocksize   = 4096         (4KB)
         read_skip        = off
         read_skipsize    = 0   (0B)

         write_random     = on
         write_size       = 1048576      (1MB)
         fsync_file       = 0
         write_blocksize  = 4096         (4KB)
         wait time        = 0

         op weights
                       read = 100 (100.00%)
                    readall = 0 (0.00%)
                      write = 0 (0.00%)
                     create = 0 (0.00%)
                     append = 0 (0.00%)
                     delete = 0 (0.00%)
                     metaop = 0 (0.00%)
                  createdir = 0 (0.00%)
                       stat = 0 (0.00%)
                   writeall = 0 (0.00%)
             writeall_fsync = 0 (0.00%)
                 open_close = 0 (0.00%)
                write_fsync = 0 (0.00%)
               create_fsync = 0 (0.00%)
               append_fsync = 0 (0.00%)

FileSystem /mnt/noorsh/ffsb/983bd7d3-a8a1-4122-95e4-f90c0ad81d5a
==========
         num_dirs         = 100
         starting files   = 64

         min file size    = 16777216     (16MB)
         max file size    = 16777216     (16MB)
         directio         = on
         alignedio        = on
         bufferedio       = off

         aging is off
         current utilization = 15.95%
```

```
checking existing fs: /mnt/noorsh/ffsb/983bd7d3-a8a1-4122-95e4-f90c0ad81d5a
fs setup took 0 secs
Syncing()...0 sec
Starting Actual Benchmark At: Mon Oct 28 14:46:13 2013

Syncing()...0 sec
FFSB benchmark finished   at: Mon Oct 28 14:47:15 2013

Results:
Benchmark took 61.55 sec

Total Results
===============
             Op Name   Transactions      Trans/sec      % Trans     % Op
                 Weight    Throughput
             =======   ============      =========      =======
                 ===========    ==========
                read :        99072        1609.66      100.000%
                           100.000%           6.29MB/sec
-
1609.66 Transactions per Second

Throughput Results
===================
Read Throughput: 6.29MB/sec

System Call Latency statistics in millisecs
=====
                Min             Avg             Max             Total Calls
                ========        ========        ========
                    ============
[   open]       0.004000        0.015760        0.052000
   387
   -
[   read]       0.163000        6.148361        228.093000
   99072
   -
[  lseek]       0.000000        0.001050        0.049000
   99072
   -
[  close]       0.002000        0.006049        0.014000
   387
   -

Discrete overall System Call Latency statistics in millisecs
=====

Overall Calls: 198918
=====
Values[ms]:
====
[   open]       Total calls:          387

0.012000
0.020000
(...)
(Single Value Output)
(...)
====
[   stat]       Total calls:            0
```

```
0.2% User   Time
2.5% System Time
2.7% CPU Utilization
```

The log output of SPA will look like this (abridged), interesting are mostly the first and the last few lines:

```
14:47:13.178 [main      ] DEBUG e.k.s.s.d.s.SQLiteDataStore    - Opening
    Database 'exampleFFSBon1Host.sqlite'
14:47:13.208 [SQLiteQueu] DEBUG e.k.s.s.d.s.SBHSQLiteQueue     - Starting
    executing Job SQLiteDataStore$1@6f878144
14:47:13.211 [SQLiteQueu] DEBUG e.k.s.s.d.s.SBHSQLiteQueue     - Finished
    executing Job SQLiteDataStore$1@6f878144
14:47:13.225 [main      ] DEBUG e.k.s.s.BenchmarkController    - Reading
    Configuration from configs/examples/exampleFFSBon1Host.xmi
14:47:13.759 [main      ] DEBUG e.k.s.s.BenchmarkController    - Validating
     Objects:
14:47:13.775 [main      ] DEBUG e.k.s.s.BenchmarkController    - Validating
     edu.kit.sdq.storagebenchmarkharness.SBHModel.Configuration.
    Configuration@508aeb74
(...)
(Validating Objects)
(...)
14:47:14.865 [main      ] DEBUG e.k.s.s.BenchmarkController    - Setup is
    edu.kit.sdq.storagebenchmarkharness.SBHModel.Configuration.
    ExperimentSetup@6e82254d (identifier: Example_FFSB_on_one_host,
    repeatCount: 1, repeatWarmup: false)
14:47:14.865 [main      ] DEBUG e.k.s.s.BenchmarkController    - Found
    series edu.kit.sdq.storagebenchmarkharness.SBHModel.Configuration.
    ExperimentSeries@225f1ae9 (identifier: FFSB_run)
14:47:14.866 [main      ] DEBUG e.k.s.s.BenchmarkController    - No
    connection found for SUT SUT_example, creating one
14:47:14.868 [main      ] DEBUG e.k.s.s.SSHRemoteConnection    - Creating
    remote connection to edu.kit.sdq.storagebenchmarkharness.SBHModel.
    Configuration.SystemUnderTest@3f3a0212 (identifier: SUT_example, ip:
    141.3.52.138, port: 22, user: noorsh, keyFile: /Users/qaisnoorshams/.
    ssh/id_rsa_IOMeasurements01)
14:47:14.869 [main      ] DEBUG e.k.s.s.BenchmarkController    - No
    connection set found for monitoring the SUT SUT_example, creating one
14:47:14.869 [main      ] DEBUG e.k.s.s.BenchmarkController    - Adding
    Experiments
14:47:14.875 [main      ] DEBUG e.k.s.s.BenchmarkDriver        -
    RawFileSaveDir is raw
14:47:14.908 [main      ] DEBUG e.k.s.s.b.f.FFSBenchmarkDriver - TargetDir
    is /mnt/noorsh/ffsb/
14:47:14.908 [main      ] DEBUG e.k.s.s.ExperimentSeriesHelper - Expanding
    series edu.kit.sdq.storagebenchmarkharness.SBHModel.Configuration.
    ExperimentSeries@225f1ae9 (identifier: FFSB_run)
14:47:14.908 [main      ] DEBUG e.k.s.s.ExperimentSeriesHelper - Creating
    SUTVariables
14:47:14.910 [main      ] DEBUG e.k.s.s.ExperimentSeriesHelper - Creating
    BenchmarkVariables
14:47:14.912 [main      ] DEBUG e.k.s.s.BenchmarkController    - Found 1
    Experiments in this series
14:47:14.914 [main      ] DEBUG e.k.s.s.d.s.SQLiteDataStore    - Setting up
     Database
14:47:14.914 [SQLiteQueu] DEBUG e.k.s.s.d.s.SBHSQLiteQueue     - Starting
    executing Job SQLiteDataStore$2@1343a083
14:47:14.928 [SQLiteQueu] DEBUG e.k.s.s.d.s.SQLiteHelper       - Create SQL
     ALTER TABLE ffsbIndependentVars ADD COLUMN fileSystem VARCHAR DEFAULT
    NULL;
(...)
(SQL Statements)
(...)
```

```
14:47:14.993 [SQLiteQueu] DEBUG e.k.s.s.d.s.SBHSQLiteQueue     - Finished
    executing Job SQLiteDataStore$2@1343a083
14:47:14.994 [main      ] DEBUG e.k.s.s.BenchmarkController    - Connecting
     to all SUTs:
14:47:14.999 [main      ] DEBUG e.k.s.s.SSHRemoteConnection    - Adding
    publickey /Users/qaisnoorshams/.ssh/id_rsa_IOMeasurements01
14:47:15.011 [main      ] DEBUG e.k.s.s.SSHRemoteConnection    - Connecting
     to 141.3.52.138:22
14:47:17.288 [SQLiteQueu] DEBUG e.k.s.s.d.s.SBHSQLiteQueue     - Starting
    executing Job SQLiteDataStore$3@d2f41a5
14:47:17.290 [SQLiteQueu] DEBUG e.k.s.s.d.s.SBHSQLiteQueue     - Finished
    executing Job SQLiteDataStore$3@d2f41a5
14:47:17.290 [main      ] DEBUG e.k.s.s.BenchmarkController    - Create
    connecting for monitoring
14:47:17.291 [main      ] DEBUG e.k.s.s.BenchmarkController    - Creating
    and starting Threads
14:47:17.293 [main      ] DEBUG e.k.s.s.BenchmarkController    - Waiting
    for Threads to finish
14:47:17.294 [H-SUT_exam] DEBUG e.k.s.s.BenchmarkController    -
    Configuration: edu.kit.sdq.storagebenchmarkharness.SBHModel.
    IndependentVariablesOfSut@21455cf0 (fileSystem: ext4, scheduler: NOOP)/
    edu.kit.sdq.storagebenchmarkharness.SBHModel.
    IndependentVariablesOfFFSB@50d8a1a0 (readPercentage: 100, blockSize:
    null, writeBlockSize: 4096, readBlockSize: 4096, filesetSize: 1024,
    sequentialAccess: null, sequentialRead: false, sequentialWrite: false,
    writeFsync: false, threadCount: 10, runTime: 60, warmUpTime: 30,
    fileSize: 16384, opsPerFile: 256, directIO: true, opsPerFileRead: null,
     opsPerFileWrite: null, opDelay: 0)
14:47:17.294 [H-SUT_exam] DEBUG e.k.s.s.BenchmarkController    - Waiting
    for barrier for preparation
14:47:17.294 [H-SUT_exam] DEBUG e.k.s.s.BenchmarkController    - Repeat 1/1
14:47:17.294 [H-SUT_exam] DEBUG e.k.s.s.BenchmarkController    - Preparing
    experiment
14:47:17.304 [H-SUT_exam] DEBUG e.k.s.s.SSHRemoteConnection    - Command is
     bash -l -c 'mkdir /mnt/noorsh/ffsb/983bd7d3-a8a1-4122-95e4-
    f90c0ad81d5a'
14:47:17.507 [H-SUT_exam] DEBUG e.k.s.s.b.f.FFSBenchmarkDriver - FFSB
    Configfile is /tmp/d3526247-d501-4501-90da-4b07cc4d3f07.warmup.ffsb and
     /tmp/d3526247-d501-4501-90da-4b07cc4d3f07.bench.ffsb
14:47:17.509 [H-SUT_exam] DEBUG e.k.s.s.SSHRemoteConnection    - Command is
     bash -l -c 'tee /tmp/d3526247-d501-4501-90da-4b07cc4d3f07.warmup.ffsb'
14:47:17.713 [H-SUT_exam] DEBUG e.k.s.s.SSHRemoteConnection    - Command is
     bash -l -c 'tee /tmp/d3526247-d501-4501-90da-4b07cc4d3f07.bench.ffsb'
14:47:17.921 [H-SUT_exam] DEBUG e.k.s.s.BenchmarkDriver        - Setting
    scheduler to NOOP
14:47:17.921 [H-SUT_exam] DEBUG e.k.s.s.SSHRemoteConnection    - Command is
     bash -l -c 'readlink -f `df -T -P /mnt/noorsh/ffsb/983bd7d3-a8a1
    -4122-95e4-f90c0ad81d5a | awk '\''NR>1 {printf $1}'\'''`'
14:47:18.140 [H-SUT_exam] DEBUG e.k.s.s.SSHRemoteConnection    - Command is
     bash -l -c 'cat /sys/class/block/xvdb/queue/scheduler'
14:47:18.355 [H-SUT_exam] DEBUG e.k.s.s.BenchmarkDriver        - Available
    schedulers are [[noop], anticipatory, deadline, cfq,
]
14:47:18.355 [H-SUT_exam] DEBUG e.k.s.s.BenchmarkDriver        - Scheduler
    noop is already active, doing nothing
14:47:18.356 [H-SUT_exam] DEBUG e.k.s.s.SSHRemoteConnection    - Command is
     bash -l -c 'df -T -P /mnt/noorsh/ffsb/983bd7d3-a8a1-4122-95e4-
    f90c0ad81d5a | awk '\''NR>1 {printf $2}'\'''
14:47:18.573 [H-SUT_exam] DEBUG e.k.s.s.BenchmarkDriver        - Current
    Filesystem ext4, expected ext4
14:47:18.573 [H-SUT_exam] DEBUG e.k.s.s.b.f.FFSBenchmarkDriver - FFSB
    Warmup
14:47:18.574 [H-SUT_exam] DEBUG e.k.s.s.SSHRemoteConnection    - Command is
     bash -l -c 'ffsb /tmp/d3526247-d501-4501-90da-4b07cc4d3f07.warmup.ffsb
    '
14:47:54.429 [H-SUT_exam] DEBUG e.k.s.s.BenchmarkController    - Waiting
    for start monitoring
14:47:54.429 [H-SUT_exam] DEBUG e.k.s.s.BenchmarkController    - Waiting
    for all monitors to be started
14:47:54.429 [H-SUT_exam] DEBUG e.k.s.s.BenchmarkController    - Starting
    Benchmarking
14:47:54.429 [H-SUT_exam] DEBUG e.k.s.s.b.f.FFSBenchmarkDriver - Executing
    FFSB for #1
14:47:54.430 [H-SUT_exam] DEBUG e.k.s.s.SSHRemoteConnection    - Command is
     bash -l -c 'ffsb /tmp/d3526247-d501-4501-90da-4b07cc4d3f07.bench.ffsb'
14:48:56.122 [H-SUT_exam] DEBUG e.k.s.s.b.f.FFSBenchmarkDriver - Found Main
     Heading
14:48:56.123 [H-SUT_exam] DEBUG e.k.s.s.b.f.FFSBenchmarkDriver - Found
    Heading open
14:48:56.131 [H-SUT_exam] DEBUG e.k.s.s.b.f.FFSBenchmarkDriver - Found
    Heading read
14:48:57.281 [H-SUT_exam] DEBUG e.k.s.s.b.f.FFSBenchmarkDriver - Found
    Heading write
14:48:57.282 [H-SUT_exam] DEBUG e.k.s.s.b.f.FFSBenchmarkDriver - Found
    Heading create
14:48:57.282 [H-SUT_exam] DEBUG e.k.s.s.b.f.FFSBenchmarkDriver - Found
    Heading lseek
14:48:57.994 [H-SUT_exam] DEBUG e.k.s.s.b.f.FFSBenchmarkDriver - Found
    Heading unlink
14:48:57.994 [H-SUT_exam] DEBUG e.k.s.s.b.f.FFSBenchmarkDriver - Found
    Heading close
14:48:57.997 [H-SUT_exam] DEBUG e.k.s.s.b.f.FFSBenchmarkDriver - Found
    Heading stat
14:48:58.017 [H-SUT_exam] DEBUG e.k.s.s.BenchmarkController    - Waiting
    for finishing of benchmarking
14:48:58.017 [H-SUT_exam] DEBUG e.k.s.s.BenchmarkController    - Stopping
    monitors...
14:48:58.017 [H-SUT_exam] DEBUG e.k.s.s.BenchmarkController    - Finished
    Monitoring
14:48:58.017 [H-SUT_exam] DEBUG e.k.s.s.BenchmarkController    - Waiting to
     store results
14:48:58.017 [H-SUT_exam] DEBUG e.k.s.s.BenchmarkController    - Result-
    Storing-Phase: 1 Results in database
14:48:58.018 [H-SUT_exam] DEBUG e.k.s.s.BenchmarkController    - Results
    stored
14:48:58.018 [SQLiteQueu] DEBUG e.k.s.s.d.s.SBHSQLiteQueue     - Starting
    executing Job SQLiteDataStore$4@56ee20fe
14:48:58.018 [H-SUT_exam] DEBUG e.k.s.s.BenchmarkController    - Finishing
    Experiment
14:48:58.018 [H-SUT_exam] DEBUG e.k.s.s.BenchmarkController    - Waiting
    for finish
14:48:58.018 [H-SUT_exam] DEBUG e.k.s.s.BenchmarkController    - Finishing
    Experiment
14:48:58.018 [SQLiteQueu] DEBUG e.k.s.s.d.s.SQLiteDataStore    - Starting
    saving 99075 results for host SUT_example, expNo 0, repeatNo 1
14:48:58.018 [H-SUT_exam] DEBUG e.k.s.s.SSHRemoteConnection    - Command is
     bash -l -c 'rm /tmp/d3526247-d501-4501-90da-4b07cc4d3f07.bench.ffsb'
14:48:58.220 [H-SUT_exam] DEBUG e.k.s.s.SSHRemoteConnection    - Command is
     bash -l -c 'rm /tmp/d3526247-d501-4501-90da-4b07cc4d3f07.warmup.ffsb'
14:48:58.421 [H-SUT_exam] DEBUG e.k.s.s.SSHRemoteConnection    - Command is
     bash -l -c 'rm -r /mnt/noorsh/ffsb/983bd7d3-a8a1-4122-95e4-
    f90c0ad81d5a'
14:48:58.822 [main      ] DEBUG e.k.s.s.BenchmarkController    - All
    Threads for expNo 0 finished
14:48:58.822 [main      ] DEBUG e.k.s.s.BenchmarkController    - Finishing
    Cofiguration Run
14:48:59.050 [SQLiteQueu] DEBUG e.k.s.s.d.s.SQLiteDataStore    - Finished
    saving 99075 results for host SUT_example, expNo 0, repeatNo 1
```

```
14:48:59.050 [SQLiteQueu] DEBUG e.k.s.s.d.s.SBHSQLiteQueue     - Finished
    executing Job SQLiteDataStore$4@56ee20fe
14:48:59.050 [SQLiteQueu] DEBUG e.k.s.s.d.s.SBHSQLiteQueue     - Starting
    executing Job SQLiteDataStore$5@32b3a5a0
14:48:59.051 [main      ] DEBUG e.k.s.s.BenchmarkController    - Closing
    all connections
14:48:59.051 [SQLiteQueu] DEBUG e.k.s.s.d.s.SBHSQLiteQueue     - Finished
    executing Job SQLiteDataStore$5@32b3a5a0
14:48:59.053 [main      ] DEBUG e.k.s.s.BenchmarkController    - Closing
    all remaining connections
14:48:59.054 [main      ] DEBUG e.k.s.s.BenchmarkController    - Shutting
    down the threadpool
14:48:59.054 [main      ] DEBUG e.k.s.s.BenchmarkController    - Closing
    the datastore
```

8.5 R Libraries

After all the required R packages are installed, cf. Section 7.1, the experiment results can be analyzed in R by loading the libraries and using the corresponding interface method (output abridged):

```
 i43pc92:StorageBenchmarkHarness qaisnoorshams$ r
WARNING: ignoring environment value of R_HOME

R version 3.0.1 (2013-05-16) -- "Good Sport"
(...)
> getwd()
[1] "/Users/qaisnoorshams/Desktop/EclipseModelingTools/workspace/
    StorageBenchmarkHarness"
> source("../RScripts/SPA.r")
data.table 1.8.10  For help type: help("data.table")
(...)
Registering doMC parallel backend
> dat = getAllFFSBVars("exampleFFSBon1Host.sqlite",type=
    SPATYPECONSTANTS$mean)
> dat
   runId              crIdentifier              crTime repeatNo expNo
1:     1 Example_FFSB_on_one_host 2013-10-28 13:47:17        1     0
2:     1 Example_FFSB_on_one_host 2013-10-28 13:47:17        1     0
        hostId                               expUid fileSystem scheduler
1: SUT_example 2843dc71-fa5f-4098-b3ac-7b03129c58c6       ext4      NOOP
2: SUT_example 2843dc71-fa5f-4098-b3ac-7b03129c58c6       ext4      NOOP
   readPercentage writeBlockSize readBlockSize filesetSize sequentialRead
1:            100           4096          4096        1024              0
2:            100           4096          4096        1024              0
   sequentialWrite writeFsync threadCount runTime warmUpTime fileSize
1:               0          0          10      60         30    16384
2:               0          0          10      60         30    16384
   opsPerFile directIO opDelay dvId benchPrefix valueId operation
       opMetric
1:        256        1       0    1        ffsb       1      read
    throughput
2:        256        1       0    1        ffsb       2      read
    responseTime
    opValue opType
1: 6.290000   mean
2: 6.148361   mean
>
>
> dat[,list(opMetric,opValue)]
       opMetric  opValue
1:   throughput 6.290000
2: responseTime 6.148361
```

```
>
> datSingle = getAllFFSBVars("exampleFFSBon1Host.sqlite",type=
    SPATYPECONSTANTS$singleValue)
> summary(datSingle[opMetric=='responseTime',opValue])
   Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
  0.163    0.705    3.873    6.148    8.101  228.100
>
```

The last commands show the results in brief, cf. benchmark log output above. One caveat is that the units are not transferred automatically and need to be considered while analyzing the results.

A minimal example for the SPA modeling functions is shown in Section 7.3.

9. Questions & Answers

Q1 — Why are forks of the benchmarks used instead of the official versions?

> **A** — Both used benchmarks include bugfixes and FFSB also includes the extensions to capture single request measurements instead of just mean values. This allows, e.g., to also analyze the distribution of the measurements.

Q2 — Why does Filebench get restarted multiple times during the benchmark execution?

> **A** — On several machines, Filebench seems to run unstably or freeze during the execution. The Benchmark Driver of Filebench is able to recognize a failed execution and restarts the benchmark automatically. This is why the execution of Filebench requires some root rights. This might result in a longer benchmarking period, but no manual interaction is required.

Q3 — What is the minimal setup to test the tool?

> **A** — You need i) a password-less connection to the system you want to benchmark (using RSA) and ii) one benchmark installed (the executable needs to reside in the user path), cf. Section 3. Root rights are not required for FFSB (only if the I/O scheduler of the system needs to be changed). A minimal configuration contains an Experiment Setup with one Experiment Series specifying i) a System Under Test (SUT), ii) an Independent Variable Space of the SUT, and iii) an Independent Variable Space of the Benchmark, cf. Figure 5.1. A minimal running example is shown in Section 8.

Q4 — Why is/are the device/s on the System Under Test getting full?

> **A** — Additionally to the log files, temporary files are stored in `/tmp`. If the experiments are interrupted due to errors, the files might not get cleaned up properly. Check the measurement and monitoring targets as well as `/tmp` for temporary files in case of interrupted runs.

Bibliography

[NBKR13] Q. Noorshams, D. Bruhn, S. Kounev, and R. Reussner, "Predictive Performance Modeling of Virtualized Storage Systems using Optimized Statistical Regression Techniques," in *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering (ICPE'13), Prague, Czech Republic, April 21–24*. New York, NY, USA: ACM, 2013.

[NBR$^+$14] Q. Noorshams, A. Busch, A. Rentschler, D. Bruhn, S. Kounev, P. Tůma, and R. Reussner, "Automated Modeling of I/O Performance and Interference Effects in Virtualized Storage Systems," in *34th IEEE International Conference on Distributed Computing Systems Workshops (ICDCS 2014 Workshops). 4th International Workshop on Data Center Performance, DCPerf '14*, 2014.